Under-approximating Cut Sets for Reachability in Large Scale Automata Networks

Loïc Paulevé[1], Geoffroy Andrieux[2], Heinz Koeppl[1]

[1] ETH Zürich, Switzerland.
[2] IRISA Rennes, France.

Abstract. In the scope of discrete finite-state models of interacting components, we present a novel algorithm for identifying sets of local states of components whose activity is necessary for the reachability of a given local state. If all the local states from such a set are disabled in the model, the concerned reachability is impossible.
Those sets are referred to as cut sets and are computed from a particular abstract causality structure, so-called Graph of Local Causality, inspired from previous work and generalised here to finite automata networks. The extracted sets of local states form an under-approximation of the complete minimal cut sets of the dynamics: there may exist smaller or additional cut sets for the given reachability.
Applied to qualitative models of biological systems, such cut sets provide potential therapeutic targets that are proven to prevent molecules of interest to become active, up to the correctness of the model. Our new method makes tractable the formal analysis of very large scale networks, as illustrated by the computation of cut sets within a Boolean model of biological pathways interactions gathering more than 9000 components.

1 Introduction

With the aim of understanding and, ultimately, controlling physical systems, one generally constructs dynamical models of the known interactions between the components of the system. Because parts of those physical processes are ignored or still unknown, dynamics of such models have to be interpreted as an over-approximation of the real system dynamics: any (observed) behaviour of the real system has to have a matching behaviour in the abstract model, the converse being potentially false. In such a setting, a valuable contribution of formal methods on abstract models of physical systems resides in the ability to prove the impossibility of particular behaviours.

Given a discrete finite-state model of interacting components, such as an automata network, we address here the computation of sets of local states of components that are necessary for reaching a local state of interest from a partially determined initial global state. Those sets are referred to as *cut sets*. Informally, each trace leading to the reachability of interest has to involve, at one point, at least one local state of a cut set. Hence, disabling in the model all the local states referenced in one cut set should prevent the occurrence of the concerned reachability from delimited initial states.

Applied to a model of a biological system where the reachability of interest is known to occur, such cut sets provide potential coupled therapeutic targets to control the activity of a particular molecule (for instance using gene knock-in/out). The contrary implies that the abstract model is not an over-approximation of the concrete system.

Contribution. In this paper, we present a new algorithm to extract sets of local states that are necessary to achieve the concerned reachability within a finite automata network. Those sets are referred to as cut sets, and we limit ourselves to N-sets, *i.e.* having a maximum cardinality of N.

The finite automata networks we are considering are closely related to 1-safe Petri nets [1] having mutually exclusive places. They subsume Boolean and discrete networks [2,3,4,5], synchronous or asynchronous, that are widely used for the qualitative modelling of biological interaction networks.

A naive, but complete, algorithm could enumerate all potential candidate N-sets, disable each of them in the model, and then perform model-checking to verify if the targeted reachability is still verified. If not, the candidate N-set is a cut set. This would roughly leads to m^N tests, where m is the total number of local states in the automata network. Considering that the model-checking within automata networks is PSPACE-complete [6], this makes such an approach intractable on large networks.

The proposed algorithm aims at being tractable on systems composed of a very large number of interacting components, but each of them having a small number of local states. Our method principally overcomes two challenges: prevent a complete enumeration of candidate N-sets; and prevent the use of model-checking to verify if disabling a set of local states break the concerned reachability. It inherently handles partially-determined initial states: the resulting cut N-set of local states are proven to be necessary for the reachability of the local state of interest from *any* of the supplied global initial states.

The computation of the cut N-sets takes advantage of an abstraction of the formal model which highlights some steps that are necessary to occur prior to the verification of a given reachability properties. This results in a causality structure called a *Graph of Local Causality* (GLC), which is inspired by [7], and that we generalise here to automata networks. Such a GLC has a size polynomial with the total number of local states in the automata network, and exponential with the number of local states within one automata. Given a GLC, our algorithm propagates and combines the cut N-sets of the local states referenced in this graph by computing unions or products, depending on the disjunctive or conjunctive relations between the necessary conditions for their reachability. The algorithm is proven to converge in the presence of dependence cycles.

In order to demonstrate the scalability of our approach, we perform the search for cut N-sets of processes within a very large Boolean model of biological processes relating more than 9000 components. Despite the highly combinatorial dynamics, a prototype implementation manages to compute up to the cut 5-sets within a few minutes. To our knowledge, this is the first time such a formal dynamical analysis has been performed on such a large dynamical model of biological system.

Related work and limitations. Cut sets are commonly defined upon graphs as set of edges or vertices which, if removed, disconnect a given pair of nodes [8]. For our purpose, this approach could be directly applied to the global transition graph (union of all traces) to identify local states or transitions for which the remove would disconnect initial states from the targeted states. However, the combinatorial explosion of the state space would make it intractable for large interacting systems.

The aim of the presented method is somehow similar to the generation of minimal cut sets in fault trees [9,10] used for reliability analysis, as the structure representing reachability causality contains both *and* and *or* connectors. However, the major difference is that we are here dealing with cyclic directed graphs which prevents the above mentioned methods to be straightforwardly applied.

Klamt *et al.* have developed a complete method for identifying minimal cut sets (also called intervention sets) dedicated to biochemical reactions networks, hence involving cycling dependencies [11]. This method has been later generalised to Boolean models of signalling networks [12]. Those algorithms are mainly based on the enumeration of possible candidates, with techniques to reduce the search space, for instance by exploiting symmetry of dynamics. Whereas intervention sets of [11,12] can contain either local states or reactions, our cut sets are only composed of local states.

Our method follows a different approach than [11,12] by not relying on candidate enumeration but computing the cut sets directly on an abstract structure derived statically from the model, which should make tractable the analysis of very large networks. The comparison with [12] is detailed in Subsect. 4.1.

In addition, our method is generic to any automata network, but relies on an abstract interpretation of dynamics which leads to under-approximating the cut sets for reachability: by ignoring certain dynamical constraints, the analysis can miss several cut sets and output cut sets that are not minimal for the concrete model. Finally, although we focus on finding the cut sets for the reachability of only *one* local state, our algorithm computes the cut sets for the (independent) reachability of all local states referenced in the GLC.

Outline. Sect. 2 introduces a generic characterisation of the *Graph of Local Causality* with respect to automata networks; Sect. 3 states and sketches the proof of the algorithm for extracting a subset of N-sets of local states necessary for the reachability of a given local state. Sect. 4 discusses the application to systems biology by comparing with the related work and applying our new method to a very large scale model of biological interactions. Finally, Sect. 5 discusses the results presented and some of their possible extensions.

Notations. $\wedge$ and $\vee$ are the usual logical *and* and *or* connectors. $[1;n] = \{1, \cdots, n\}$. Given a finite set A, $\#A$ is the cardinality of A; $\wp(A)$ is the power set of A; $\wp^{\leq N}(A)$ is the set of all subsets of A with cardinality at most N. Given sets $A^1, \ldots, A^n$, $\bigcup_{i \in [1;n]} A^i$ is the union of those sets, with the empty union $\bigcup_{\emptyset} \overset{\Delta}{=} \emptyset$; and $A^1 \times \cdots \times A^n$ is the usual Cartesian product. Given sets of sets $B^1, \ldots, B^n \in \wp(\wp(A))$, $\widetilde{\prod}_{i \in [1;n]} B^i \overset{\Delta}{=} B^1 \tilde{\times} \cdots \tilde{\times} B^n \in \wp(\wp(A))$ is the *sets of sets product* where $\{e_1, \ldots, e_n\} \tilde{\times} \{e'_1, \ldots, e'_m\} \overset{\Delta}{=} \{e_i \cup e'_j \mid i \in [1;n] \wedge j \in [1;m]\}$. In particular $\forall (i,j) \in [1;n] \times [1;m]$, $B^i \tilde{\times} B^j = B^j \tilde{\times} B^i$ and $\emptyset \tilde{\times} B^i = \emptyset$. The empty sets of sets product $\widetilde{\prod}_{\emptyset} \overset{\Delta}{=} \{\emptyset\}$. If $M : A \mapsto B$ is a mapping from elements in A to elements in B, $M(a)$ is the value in B mapped to $a \in A$; $M\{a \mapsto b\}$ is the mapping M where $a \in A$ now maps to $b \in B$.

2 Graph of Local Causality

We first give basic definitions of automata networks, local state disabling, context and local state reachability; then we define the local causality of an objective (local reachability), and the *Graph of Local Causality*. A simple example is given at the end of the section.

2.1 Finite Automata Networks

We consider a network of automata $(\Sigma, S, \mathcal{L}, T)$ which relates a finite number of interacting finite state automata Σ (Def. 1). The global state of the system is the gathering of the local state of composing automata. A transition can occur if and only if all the local states sharing a common transition label $\ell \in L$ are present in the global state $s \in S$ of the system. Such networks characterize a class of 1-safe Petri Nets [1] having groups of mutually exclusive places, acting as the automata. They allow the modelling of Boolean networks and their discrete generalisation, having either synchronous or asynchronous transitions.

Definition 1 (Automata Network $(\Sigma, S, \mathcal{L}, T)$). *An automata network is defined by a tuple $(\Sigma, S, \mathcal{L}, T)$ where*

- *$\Sigma = \{a, b, \ldots, z\}$ is the finite set of automata identifiers;*
- *For any $a \in \Sigma$, $S(a) = [1; k_a]$ is the finite set of local states of automaton a; $S = \prod_{a \in \Sigma} [1; k_a]$ is the finite set of global states.*
- *$\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$ is the finite set of transition labels;*
- *$T = \{a \mapsto T_a \mid a \in \Sigma\}$, where $\forall a \in \Sigma, T_a \subset [1; k_a] \times \mathcal{L} \times [1; k_a]$, is the mapping from automata to their finite set of local transitions.*

We note $i \xrightarrow{\ell} j \in T(a) \overset{\Delta}{\Leftrightarrow} (i, \ell, j) \in T_a$ and $a_i \xrightarrow{\ell} a_j \in T \overset{\Delta}{\Leftrightarrow} i \xrightarrow{\ell} j \in T(a)$.
$\forall \ell \in \mathcal{L}$, we note ${}^\bullet\ell \overset{\Delta}{=} \{a_i \mid a_i \xrightarrow{\ell} a_j \in T(a)\}$ and $\ell^\bullet \overset{\Delta}{=} \{a_j \mid a_i \xrightarrow{\ell} a_j \in T(a)\}$.
The set of local states is defined as $\mathbf{LS} \overset{\Delta}{=} \{a_i \mid a \in \Sigma \wedge i \in [1; k_a]\}$.
The global transition relation $\rightarrow \subset S \times S$ is defined as:

$$s \rightarrow s' \overset{\Delta}{\Leftrightarrow} \exists \ell \in \mathcal{L} : \forall a_i \in {}^\bullet\ell, s(a) = a_i \wedge \forall a_j \in \ell^\bullet, s'(a) = a_j$$
$$\wedge \forall b \in \Sigma, S(b) \cap {}^\bullet\ell = \emptyset \Rightarrow s(b) = s'(b).$$

Given an automata network $\mathcal{S}ys = (\Sigma, S, \mathcal{L}, T)$ and a subset of its local states $ls \subseteq \mathbf{LS}$, $\mathcal{S}ys \ominus ls$ refers to the system where all the local states ls have been disabled, i.e. they can not be involved in any transition (Def. 2).

Definition 2 (Process Disabling). *Given $\mathcal{S}ys = (\Sigma, S, \mathcal{L}, T)$ and $ls \in \wp(\mathbf{LS})$, $\mathcal{S}ys \ominus ls \overset{\Delta}{=} (\Sigma, S, \mathcal{L}', T')$ where $\mathcal{L}' = \{\ell \in \mathcal{L} \mid ls \cap {}^\bullet\ell = \emptyset\}$ and $T' = \{a_i \xrightarrow{\ell} a_j \in T \mid \ell \in \mathcal{L}'\}$.*

From a set of acceptable initial states delimited by a *context* ς (Def. 3), we say a given local state $a_j \in \mathbf{LS}$ is reachable if and only if there exists a finite number of transitions in $\mathcal{S}ys$ leading to a global state where a_j is present (Def. 4).

Definition 3 (Context ς). *Given a network $(\Sigma, S, \mathcal{L}, T)$, a context ς is a mapping from each automaton $a \in \Sigma$ to a non-empty subset of its local states: $\forall a \in \Sigma, \varsigma(a) \in \wp(S(a)) \wedge \varsigma(a) \neq \emptyset$.*

Definition 4 (Process reachability). *Given a network $(\Sigma, S, \mathcal{L}, T)$ and a context ς, the local state $a_j \in \mathbf{LS}$ is* reachable *from ς if and only if $\exists s_0, \dots, s_m \in S$ such that $\forall a \in \Sigma, s_0(a) \in \varsigma(a)$, and $s_0 \rightarrow \cdots \rightarrow s_m$, and $s_m(a) = j$.*

2.2 Local Causality

Locally reasoning within one automaton a, the global reachability of a_j from ς can be expressed as the reachability of a_j from a local state $a_i \in \varsigma(a)$. This local reachability specification is referred to as an *objective* noted $a_i \to^* a_j$ (Def. 5)

Definition 5 (Objective). *Given a network $(\Sigma, S, \mathcal{L}, T)$, the reachability of local state a_j from a_i is called an* objective *and is denoted $a_i \to^* a_j$. The set of all objectives is referred to as $\mathbf{Obj} \overset{\Delta}{=} \{a_i \to^* a_j \mid (a_i, a_j) \in \mathbf{LS} \times \mathbf{LS}\}$.*

Given an objective $P = a_i \to^* a_j \in \mathbf{Obj}$, we define $\mathsf{sol}(P)$ the *local causality* of P (Def. 6): each element $ls \in \mathsf{sol}(P)$ is a subset of local states, referred to as a (local) solution for P, which are all involved at some times prior to the reachability of a_j. $\mathsf{sol}(P)$ is sound as soon as the disabling of at least one local state in *each* solution makes the reachability of a_j impossible from any global state containing a_i (Property 1). Note that if $\mathsf{sol}(P) = \{\{a_i\} \cup ls^1, \dots, ls^m\}$ is sound, $\mathsf{sol}'(P) = \{ls^1, \dots, ls^m\}$ is also sound. $\mathsf{sol}(a_i \to^* a_j) = \emptyset$ implies that a_j can never be reached from a_i, and $\forall a_i \in \mathbf{LS}, \mathsf{sol}(a_i \to^* a_i) \overset{\Delta}{=} \{\emptyset\}$.

Definition 6. $\mathsf{sol} : \mathbf{Obj} \mapsto \wp(\wp(\mathbf{LS}))$ *is a mapping from objectives to sets of sets of local states such that $\forall P \in \mathbf{Obj}, \forall ls \in \mathsf{sol}(P), \nexists ls' \in \mathsf{sol}(P), ls \neq ls'$ such that $ls' \subset ls$. The set of these mappings is noted $\mathbf{Sol} \overset{\Delta}{=} \{\langle P, ls\rangle \mid ls \in \mathsf{sol}(P)\}$.*

Property 1 (sol *soundness).* $\mathsf{sol}(a_i \to^* a_j) = \{ls^1, \dots, ls^n\}$ is a sound set of solutions for the network $\mathcal{S}ys = (\Sigma, S, \mathcal{L}, T)$ if and only if $\forall kls \in \widetilde{\prod}_{i \in [1;n]} ls^i$, a_j is not reachable in $\mathcal{S}ys \ominus kls$ from any state $s \in S$ such that $s(a) = i$.

In the rest of this paper we assume that Property 1 is verified, and consider sol computation out of the scope of this paper.

Nevertheless, we briefly describe a construction of a sound $\mathsf{sol}(a_i \to^* a_j)$ for an automata network $(\Sigma, S, \mathcal{L}, T)$; an example is given at the end of this section. This construction generalises the computation of GLC from the Process Hitting framework, a restriction of network of automata depicted in [7]. For each acyclic sequence $a_i \xrightarrow{\ell_1} \dots \xrightarrow{\ell_m} a_j$ of local transitions in $T(a)$, and by defining $\mathsf{ext}_a(\ell) \overset{\Delta}{=} \{b_j \in \mathbf{LS} \mid b_j \xrightarrow{\ell} b_k \in T, b \neq a\}$, we set $ls \in \widetilde{\prod}_{\ell \in \{\ell_1,\dots,\ell_m \mid \mathsf{ext}_a(\ell) \neq \emptyset\}} \mathsf{ext}_a(\ell) \Rightarrow ls \in \mathsf{sol}(a_i \to^* a_j)$, up to sursets removing. One can easily show that Property 1 is verified with such a construction. The complexity of this construction is exponential in the number of local states of automata and polynomial in the number of automata. Alternative constructions may also provide sound (and not necessarily equal) sol.

2.3 Graph of Local Causality

Given a local state $a_j \in \mathbf{LS}$ and an initial context ς, the reachability of a_i is equivalent to the realization of any objective $a_i \to^* a_j$, with $a_i \in \varsigma(a)$. By definition, if a_j is reachable from ς, there exists $ls \in \mathsf{sol}(a_i \to^* a_j)$ such that, $\forall b_k \in ls$, b_k is reachable from ς.

The (directed) *Graph of Local Causality* (GLC, Def. 7) relates this recursive reasoning from a given set of local states $\omega \subseteq \mathbf{LS}$ by linking every local state a_j to all objectives $a_i \to^* a_j, a_i \in \varsigma(a)$; every objective P to its solutions $\langle P, ls\rangle \in \mathbf{Sol}$; every solution $\langle P, ls\rangle$ to its local states $b_k \in ls$. A GLC is said to be valid if sol is sound for all referenced objectives (Property 2).

Definition 7 (Graph of Local Causality). *Given a context ς and a set of local states $\omega \subseteq \mathbf{LS}$, the* Graph of Local Causality *(GLC) $\mathcal{A}^\omega_\varsigma \overset{\Delta}{=} (V^\omega_\varsigma, E^\omega_\varsigma)$, with $V^\omega_\varsigma \subseteq \mathbf{LS} \cup \mathbf{Obj} \cup \mathbf{Sol}$ and $E^\omega_\varsigma \subseteq V^\omega_\varsigma \times V^\omega_\varsigma$, is the smallest structure satisfying:*

$$\begin{aligned}
\omega &\subseteq V^\omega_\varsigma \\
a_i \in V^\omega_\varsigma \cap \mathbf{LS} &\Leftrightarrow \{(a_i, a_j \to^* a_i) \mid a_j \in \varsigma\} \subseteq E^\omega_\varsigma \\
a_i \to^* a_j \in V^\omega_\varsigma \cap \mathbf{Obj} &\Leftrightarrow \{(a_i \to^* a_j, \langle a_i \to^* a_j, ls\rangle) \mid \langle a_i \to^* a_j, ls\rangle \in \mathbf{Sol}\} \subseteq E^\omega_\varsigma \\
\langle P, ls\rangle \in V^\omega_\varsigma \cap \mathbf{Sol} &\Leftrightarrow \{(\langle P, ls\rangle, a_i) \mid a_i \in ls\} \subseteq E^\omega_\varsigma \quad .
\end{aligned}$$

Property 2 (Valid Graph of Local Causality). A GLC $\mathcal{A}^\omega_\varsigma$ is *valid* if, $\forall P \in V^\omega_\varsigma \cap \mathbf{Obj}$, $\mathsf{sol}(P)$ is sound.

This structure can be constructed starting from local states in ω and by iteratively adding the imposed children. It is worth noticing that this graph can contain cycles. In the worst case, $\#V^\omega_\varsigma = \#\mathbf{LS} + \#\mathbf{Obj} + \#\mathbf{Sol}$ and $\#E^\omega_\varsigma = \#\mathbf{Obj} + \#\mathbf{Sol} + \sum_{\langle P, ls\rangle \in \mathbf{Sol}} \#ls$.

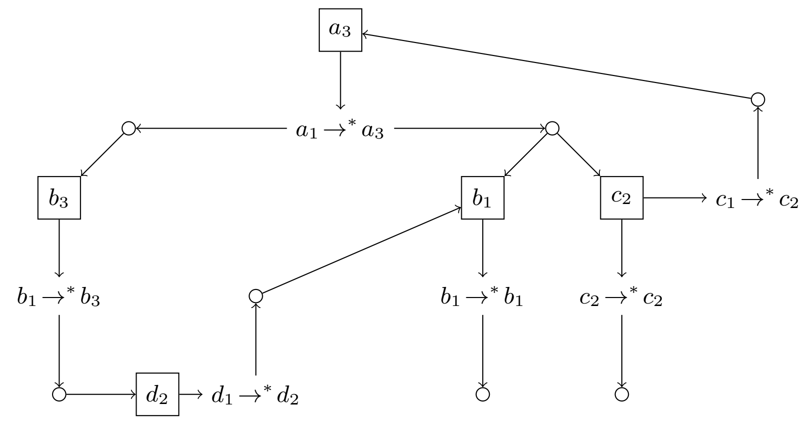

Fig. 1. Example of Graph of Local Causality that is valid for the automata network defined in Example 1

Example 1. Fig. 1 shows an example of GLC. Local states are represented by boxed nodes and elements of **Sol** by small circles.

For instance, such a GLC is valid for the following automata network $(\Sigma, S, \mathcal{L}, T)$, with initial context $\varsigma = \{a \mapsto \{1\}; b \mapsto \{1\}; c \mapsto \{1,2\}; d \mapsto \{2\}\}$:

$$\begin{aligned}
\Sigma &= \{a, b, c, d\} & \mathcal{L} &= \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6\} \\
S(a) &= [1; 3] & T(a) &= \{1 \xrightarrow{\ell_2} 2; 2 \xrightarrow{\ell_3} 3; 1 \xrightarrow{\ell_1} 3; 3 \xrightarrow{\ell_4} 2\} \\
S(b) &= [1; 3] & T(b) &= \{1 \xrightarrow{\ell_2} 2; 1 \xrightarrow{\ell_5} 3; 1 \xrightarrow{\ell_6} 1; 3 \xrightarrow{\ell_1} 2\} \\
S(c) &= [1; 2] & T(c) &= \{1 \xrightarrow{\ell_4} 2; 2 \xrightarrow{\ell_3} 1\} \\
S(d) &= [1; 2] & T(d) &= \{1 \xrightarrow{\ell_6} 2; 2 \xrightarrow{\ell_5} 1\}
\end{aligned}$$

For example, within automata a, there are two acyclic sequences from 1 to 3: $1 \xrightarrow{\ell_2} 2 \xrightarrow{\ell_3} 3$ and $1 \xrightarrow{\ell_1} 3$. Hence, if a_3 is reached from a_1, then necessarily, one of these two sequences has to be used (but not necessarily consecutively). For each of these transitions, the transition label is shared by exactly one local state in another automaton: b_1, c_2, b_3 for ℓ_2, ℓ_3, ℓ_1, respectively. Therefore, if a_3 is reached from a_1, then necessarily either both b_1 and c_2, or b_3 have been reached before. Hence $\mathsf{sol}(a_1 \rightarrow^* a_3) = \{\{b_1, c_2\}, \{b_3\}\}$ is sound, as disabling either b_1 and b_3, or c_2 and b_3, would remove any possibility to reach a_3 from a_1.

3 Necessary Processes for Reachability

We assume a global sound GLC $\mathcal{A}_\varsigma^\omega = (V_\varsigma^\omega, E_\varsigma^\omega)$, with the usual accessors for the direct relations of nodes:

$$\mathsf{children} : V_\varsigma^\omega \mapsto \wp(V_\varsigma^\omega) \qquad \mathsf{parents} : V_\varsigma^\omega \mapsto \wp(V_\varsigma^\omega)$$

$$\mathsf{children}(n) \triangleq \{m \in V_\varsigma^\omega \mid (n,m) \in E_\varsigma^\omega\} \quad \mathsf{parents}(n) \triangleq \{m \in V_\varsigma^\omega \mid (m,n) \in E_\varsigma^\omega\}$$

Given a set of local states $\mathcal{O}bs \subseteq \mathbf{LS}$, this section introduces an algorithm computing upon $\mathcal{A}_\varsigma^\omega$ the set $\mathbb{V}(a_i)$ of minimal cut N-sets of local states in $\mathcal{O}bs$ that are necessary for the independent reachability of each local state $a_i \in \mathbf{LS} \cap V_\varsigma^\omega$. The minimality criterion actually states that $\forall ls \in \mathbb{V}(a_i)$, there is no different $ls' \in \mathbb{V}(a_i)$ such that $ls' \subset ls$.

Assuming a first valuation $\mathbb{V}$ (Def. 8) associating to each node a set of (minimal) cut N-sets, the set of cut N-sets for the node n can be refined following $\mathsf{update}(\mathbb{V}, n)$ (Def. 9):

- if n is a solution $\langle P, ls\rangle \in \mathbf{Sol}$, it is sufficient to prevent the reachability of *any* local state in ls to cut n; therefore, the cut N-sets results from the union of the cut N-sets of n children (all local states).
- If n is an objective $P \in \mathbf{Obj}$, all its solutions (in $\mathsf{sol}(P)$) have to be cut in order to ensure that P is not realizable: hence, the cut N-sets result from the product of children cut N-sets (all solutions).
- If n is a local state a_i, it is sufficient to cut all its children (all objectives) to prevent the reachability of a_i from any state in the context ς. In addition, if $a_i \in \mathcal{O}bs$, $\{a_i\}$ is added to the set of its cut N-sets.

Definition 8 (Valuation $\mathbb{V}$). *A* valuation $\mathbb{V} : V_\varsigma^\omega \mapsto \wp(\wp^{\leq N}(\mathcal{O}bs))$ *is a mapping from each node of $\mathcal{A}_\varsigma^\omega$ to a set of N-sets of local states.* $\mathbf{Val}$ *is the set of all valuations.* $\mathbb{V}_0 \in \mathbf{Val}$ *refers to the valuation such that* $\forall n \in V_\varsigma^\omega, \mathbb{V}_0(n) = \emptyset$.

Definition 9 ($\mathsf{update} : \mathbf{Val} \times V_\varsigma^\omega \mapsto \mathbf{Val}$).

$$\mathsf{update}(\mathbb{V}, n) \triangleq \begin{cases} \mathbb{V}\{n \mapsto \zeta^N(\bigcup_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \textit{if } n \in \mathbf{Sol} \\ \mathbb{V}\{n \mapsto \zeta^N(\widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \textit{if } n \in \mathbf{Obj} \\ \mathbb{V}\{n \mapsto \zeta^N(\widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \textit{if } n \in \mathbf{LS} \setminus \mathcal{O}bs \\ \mathbb{V}\{n \mapsto \zeta^N(\{\{a_i\}\} \cup \widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \textit{if } n \in \mathbf{LS} \cap \mathcal{O}bs \end{cases}$$

where $\zeta^N(\{e_1, \ldots, e_n\}) \triangleq \{e_i \mid i \in [1;n] \wedge \#e_i \leq N \wedge \nexists j \in [1;n], j \neq i, e_j \subset e_i\}$, e_i *being sets,* $\forall i \in [1;n]$.

Starting with $\mathbb{V}_0$, one can repeatedly apply update on each node of $\mathcal{A}_\varsigma^\omega$ to refine its valuation. Only nodes where one of their children value has been modified should be considered for updating.

Hence, the order of nodes updates should follow the topological order of the GLC, where children have a lower rank than their parents (i.e., children are

Algorithm 1 $\mathcal{A}_{\varsigma}^{\omega}$-MINIMAL-CUT-NSETS

```
1: $\mathcal{M} \leftarrow V_{\varsigma}^{\omega}$
2: $\mathbb{V} \leftarrow \mathbb{V}_0$
3: while $\mathcal{M} \neq \emptyset$ do
4:     $n \leftarrow \arg\min_{m \in \mathcal{M}}\{\mathsf{rank}(m)\}$
5:     $\mathcal{M} \leftarrow \mathcal{M} \setminus \{n\}$
6:     $\mathbb{V}' \leftarrow \mathsf{update}(\mathbb{V}, n)$
7:     if $\mathbb{V}'(n) \neq \mathbb{V}(n)$ then
8:         $\mathcal{M} \leftarrow \mathcal{M} \cup \mathsf{parents}(n)$
9:     end if
10:    $\mathbb{V} \leftarrow \mathbb{V}'$
11: end while
12: return $\mathbb{V}$
```

treated before their parents). If the graph is actually acyclic, then it is sufficient to update the value of each node only once. In the general case, *i.e.* in the presence of Strongly Connected Components (SCCs) — nodes belonging to the same SCC have the same rank —, the nodes within a SCC have to be iteratively updated until the convergence of their valuation.

Algorithm 1 formalizes this procedure where $\mathsf{rank}(n)$ refers to the topological rank of n, as it can be derived from Tarjan's strongly connected components algorithm [13], for example. The node $n \in V_{\varsigma}^{\omega}$ to be updated is selected as being the one having the least rank amongst the nodes to update (delimited by $\mathcal{M}$). In the case where several nodes with the same lowest rank are in $\mathcal{M}$, they can be either arbitrarily or randomly picked. Once picked, the value of n is updated. If the new valuation of n is different from the previous, the parents of n are added to the list of nodes to update (lines 6-8 in Algorithm 1).

Lemma 1 states the convergence of Algorithm 1 and Theorem 1 its correctness: for each local state $a_i \in V_{\varsigma}^{\omega} \cap \mathbf{LS}$, each set of local states $kls \in \mathbb{V}(a_i)$ (except $\{a_i\}$ singleton) references the local states that are all necessary to reach prior to the reachability of a_i from any state in ς. Hence, if all the local states in kls are disabled in $\mathcal{S}ys$, a_i is not reachable from any state in ς.

Lemma 1. $\mathcal{A}_{\varsigma}^{\omega}$-MINIMAL-CUT-NSETS *always terminates.*

Proof. Remarking that $\wp(\wp^{\leq N}(\mathcal{O}bs))$ is finite, defining a partial ordering such that $\forall v, v' \in \wp(\wp^{\leq N}(\mathcal{O}bs)), v \succeq v' \overset{\Delta}{\Leftrightarrow} \zeta^N(v) = \zeta^N(v \cup v')$, and noting $\mathbb{V}^k \in \mathbf{Val}$ the valuation after k iterations of the algorithm, it is sufficient to prove that $\mathbb{V}^{k+1}(n) \succeq \mathbb{V}^k(n)$. Let us define $v_1, v_2, v_1', v_2' \in \wp(\wp^{\leq N}(\mathcal{O}bs))$ such that $v_1 \succeq v_1'$ and $v_2 \succeq v_2'$. We can easily check that $v_1 \cup v_2 \succeq v_1' \cup v_2'$ (hence proving the case when $n \in \mathbf{Sol}$). As $\zeta^N(v_1) = \zeta^N(v_1 \cup v_1') \Leftrightarrow \forall e_1' \in v_1', \exists e_1 \in v_1 : e_1 \subseteq e_1'$, we obtain that $\forall (e_1', e_2') \in v_1' \times v_2', \exists (e_1, e_2) \in v_1 \times v_2 : e_1 \subseteq e_1' \wedge e_2 \subseteq e_2'$. Hence $e_1 \cup e_2 \subseteq e_1' \cup e_2'$, therefore $\zeta^N(v_1 \tilde{\times} v_2 \cup v_1' \tilde{\times} v_2') = \zeta^N(v_1 \tilde{\times} v_2)$, i.e. $v_1 \tilde{\times} v_2 \succeq v_1' \tilde{\times} v_2'$; which proves the cases when $n \in \mathbf{Obj} \cup \mathbf{LS}$.

Node	rank	$\mathbb{V}$
$\langle b_1 \rightarrow^* b_1, \emptyset\rangle$	1	$\emptyset$
$b_1 \rightarrow^* b_1$	2	$\emptyset$
b_1	3	$\{\{b_1\}\}$
$\langle d_1 \rightarrow^* d_2, \{b_1\}\rangle$	4	$\{\{b_1\}\}$
$d_1 \rightarrow^* d_2$	5	$\{\{b_1\}\}$
d_2	6	$\{\{b_1\}, \{d_2\}\}$
$\langle b_1 \rightarrow^* b_3, \{d_2\}\rangle$	7	$\{\{b_1\}, \{d_2\}\}$
$b_1 \rightarrow^* b_3$	8	$\{\{b_1\}, \{d_2\}\}$
b_3	9	$\{\{b_1\}, \{b_3\}, \{d_2\}\}$
$\langle a_1 \rightarrow^* a_3, \{b_3\}\rangle$	10	$\{\{b_1\}, \{b_3\}, \{d_2\}\}$
$\langle c_2 \rightarrow^* c_2, \emptyset\rangle$	11	$\emptyset$
$c_2 \rightarrow^* c_2$	12	$\emptyset$
c_2	13	$\{\{c_2\}\}$
$\langle a_1 \rightarrow^* a_3, \{b_1, c_2\}\rangle$	13	$\{\{b_1\}, \{c_2\}\}$
$a_1 \rightarrow^* a_3$	13	$\{\{b_1\}, \{b_3, c_2\}, \{c_2, d_2\}\}$
a_3	13	$\{\{a_3\}, \{b_1\}, \{b_3, c_2\}, \{c_2, d_2\}\}$
$\langle c_1 \rightarrow^* c_2, \{a_3\}\rangle$	13	$\{\{a_3\}, \{b_1\}, \{b_3, c_2\}, \{c_2, d_2\}\}$

Table 1. Result of the execution of Algorithm 1 on the GLC in Fig. 1

Theorem 1. *Given a GLC $\mathcal{A}^\omega_\varsigma = (V^\omega_\varsigma, E^\omega_\varsigma)$ which is sound for the automata network $\mathcal{S}ys$, the valuation $\mathbb{V}$ computed by $\mathcal{A}^\omega_\varsigma$-*Minimal-Cut-NSets *verifies: $\forall a_i \in \mathbf{LS} \cap V^\omega_\varsigma, \forall kls \in \mathbb{V}(a_i) \setminus \{\{a_i\}\}$, a_j is not reachable from ς within $\mathcal{S}ys \ominus kls$.*

Proof. By recurrence on the valuations $\mathbb{V}$: the above property is true at each iteration of the algorithm.

Example 2. Table 1 details the result of the execution of Algorithm 1 on the GLC defined in Fig. 1. Nodes receive a topological rank, identical ranks implying the belonging to the same SCC. The (arbitrary) scheduling of the updates of nodes within a SCC follows the order in the table. In this particular case, nodes are all visited once, as $\mathbb{V}(\langle c_2 \rightarrow^* c_2, \emptyset\rangle) \tilde{\times} \mathbb{V}(\langle c_1 \rightarrow^* c_2, \{a_3\}\rangle) = \emptyset$ (hence $\mathsf{update}(\mathbb{V}, c_2)$ does not change the valuation of c_2). Note that in general, several iterations of update may be required to reach a fixed point.

It is worth noticing that the GLC abstracts several dynamical constraints in the underlying automata networks, such as the ordering of transitions, or the synchronous updates of the global state. In that sense, GLC over-approximates the dynamics of the network, and the resulting cut sets are under-approximating the complete cut sets of the concrete model.

4 Application to Systems Biology

Automata networks, as presented in Def. 1, subsume Boolean and discrete networks, synchronous and asynchronous, that are widely used for the qualitative modelling of dynamics of biological networks [2,3,14,4,5,15,16].

A cut set, as extracted by our algorithm, informs that at least one of the component in the cut set has to be present in the specified local state in order to achieve the wanted reachability. A local state can represent, for instance, an active transcription factor or the absence of a certain protein. It provides potential therapeutic targets if the studied reachability is involved in a disease by preventing all the local states of a cut set to act, for instance using gene knock-out or knock-in techniques.

We first discuss and compare our methodology with the *intervention sets* analysis within biological models developed by S. Klamt *et al.*, and provide some benchmarks on a few examples.

Thanks to the use of the intermediate GLC and to the absence of candidate enumeration, our new method makes tractable the cut sets analysis on very large models. We present a recent application of our results to the analysis of a very large scale Boolean model of biological pathway interactions involving 9000 components. To our knowledge, this is the first attempt of a formal dynamical analysis on such a large scale model.

4.1 Related Work

The general related work having been discussed in Sect. 1, we deepen here the comparison of our method with the closest related work: the analysis of *Intervention Sets* (IS) [12]. In the scope of Boolean models of signalling networks, an IS is a set of local states such that forcing the components to stick at these local states ensures that the system always reaches a fixed point (steady state) where certain target components have the desired state. Their method is complete, i.e., all possible ISs are computed; and contrary to our, allow the specification of more than one local state for the target state of the intervention.

Nevertheless, the semantics and the computation of ISs have some key differences with our computed cut sets. First, they focus only on the reachability of (logical) steady states, which is a stronger condition than the transient reachability that we are considering. Then, the steady states are computed using a three-valued logic which allows to cope with undefined (initial) local states, but which is different from the notion of context that we use in this paper for specifying the initial condition.

Such differences make difficult a proper comparison of inferred cut sets. We can however expect that any cut sets found by our method has a corresponding IS in the scope of Boolean networks when the initial context actually specifies a single initial state.

To give a practical insight on the relation between the two methods, we compare the results for two signalling networks, both between a model specified with CellNetAnalyser [17] to compute ISs and a model specified in the Process Hitting framework, a particular restriction of asynchronous automata networks [18], to compute our cut sets. Process Hitting models have been built in order to over-approximate the dynamics considered for the computation of ISs[3].

[3] Models and scripts available at `http://loicpauleve.name/cutsets.tbz2`

Tcell. Applied to a model of the T-cell receptor signalling between 40 components [19], we are interested in preventing the activation of the transcription factor $AP1$. For an instance of initial conditions, and limiting the computations to 3-sets, 31 ISs have been identified (28 1-sets, 3 2-sets, 0 3-set), whereas our algorithm found 29 cut sets (21 1-sets and 8 2-sets), which are all matching an IS (23 are identical, 6 strictly including ISs). ISs are computed in 0.69s while our algorithm under-approximates the cut sets in 0.006s. Different initial states give comparable results.

Egfr. Applied to a model of the epidermal growth factor receptor signalling pathway of 104 components [15], we are interested in preventing the activation of the transcription factor $AP1$. For an instance of initial conditions, and limiting the computations to 3-sets, 25 ISs have been identified (19 1-sets, 3 2-sets, 3 3-sets), whereas our algorithm found 14 cut sets (14 1-sets), which are all included in the ISs. ISs are computed in 98s while our algorithm under-approximates the cut sets in 0.004s. Different initial states give comparable results.

As expected with the different semantics of models and cut sets, resulting ISs matches all the cut sets identified by our algorithm, and provides substantially more sets. The execution time is much higher for ISs as they rely on candidate enumeration in order to provide complete results, whereas our method was designed to prevent such an enumeration but under-approximates the cut sets.

In order to appreciate the under-approximation done by our method at a same level of abstraction and with identical semantics, we compare the cut sets identified by our algorithm with the cut sets obtained using a naive, but complete, computation. The naive computation enumerates all cut set candidates and, for each of them, disable the local states in the model and perform model-checking to verify if the target local state is still reachable. In the particular case of these two models, and limiting the cut sets to 3 and 2-sets respectively for the sake of tractability, no additional cut set has been uncovered by the complete method. Such a good under-approximation could be partially explained by the restrictions imposed on the causality by the Process Hitting framework, making the GLC a tight over-approximation of the dynamics [7].

4.2 Very Large Scale Application to Pathway Interactions

In order to support the scalability of our approach, we apply the proposed algorithm to a very large model of biological interactions, actually extracted from the PID database [20] referencing various influences (complex formation, inductions (activations) and inhibitions, transcriptional regulation, etc.) between more than 9000 biological components (proteins, genes, ions, etc.).

Amongst the numerous biological components, the activation of some of them are known to control key mechanisms of the cell dynamics. Those activations are the consequence of intertwining signalling pathways and depend on the environment of the cell (represented by the presence of certain *entry-point* molecules).

Model	N	Visited nodes	Exec. time	Nb. of resulting N-sets		
				SNAIL_1	p15INK4b_1	p21CIP1_1
whole_PID_OR	1	29022	0.9s	1	1	1
	2	36602	1.6s	+6	+6	+0
	3	44174	5.4s	+0	+92	+0
	4	54322	39s	+30	+60	+0
	5	68214	8.3m	+90	+80	+0
	6	90902	2.6h	+930	+208	+0

Table 2. Number of nodes visited and execution time of the search for cut N-sets of 3 local states. For each N, only the number of additional N-sets is displayed.

Uncovering the environmental and intermediate components playing a major role in these signalling dynamics is of great biological interest.

The full PID database has been interpreted into the Process Hitting framework, a subclass of asynchronous automata networks, from which the derivation of the GLC has been addressed in previous work [7]. The obtained model gathers components representing either biological entities modelled as boolean value (absent or present), or logical complexes. When a biological component has several competing regulators, the precise cooperations are not detailed in the database, so we use of two different interpretations: all (resp. one of) the activators and none (resp. all but one of) the inhibitors have to be present in order to make the target component present. This leads to two different discrete models of PID that we refer to as `whole_PID_AND` and `whole_PID_OR`, respectively.

Focusing on `whole_PID_OR`, the Process Hitting model relates more than 21000 components, either biological or logical, containing between 2 and 4 local states. Such a system could actually generate 2^{33874} states. 3136 components act as environment specification, which in our boolean interpretation leads to 2^{3136} possible initial states, assuming all other components start in the absent state.

We focus on the (independent) reachability of active SNAIL transcription factor, involved in the epithelial to mesenchymal transition [21], and of active p15INK4b and p21CIP1 cyclin-dependent kinase inhibitors involved in cell cycle regulation [22]. The GLC relates 20045 nodes, including 5671 component local states (biological or logical); it contains 6 SCCs with at least 2 nodes, the largest being composed of 10238 nodes and the others between 20 and 150.

Table 2 shows the results of a prototype implementation[4] of Algorithm 1 for the search of up to the 6-sets of biological component local states. One can observe that the execution time grows very rapidly with N compared to the number of visited nodes. This can be explained by intermediate nodes having a large set of cut N-sets leading to a costly computation of products.

While the precise biological interpretation of identified N-sets is out of the scope of this paper, we remark that the order of magnitude of the number of cut sets can be very different (more than 1000 cut 6-sets for SNAIL; none cut

[4] Implemented as part of the Pint software – `http://process.hitting.free.fr`
Models and scripts available at `http://loicpauleve.name/cutsets.tbz2`

6-sets for p21CIP1, except the gene that produces this protein). It supports a notion of robustness for the reachability of components, where the less cut sets, the more robust the reachability to various perturbations.

Applied to the `whole_PID_AND` model, our algorithm find in general much more cut N-sets, due to the conjunctive interpretation. This brings a significant increase in the execution time: the search up to the cut 5-sets took 1h, and the 6-sets leads to an out-of-memory failure.

Because of the very large number of components involved in this model, the naive exact algorithm consisting in enumerating all possible N-sets candidates and verifying the concerned reachability using model-checking is not tractable. Similarly, making such a model fit into other frameworks, such as CellNetAnalyser (see previous sub-section) is a challenging task, and might be considered as future work.

5 Discussion

We presented a new method to efficiently compute cut sets for the reachability of a local state of a component within networks of finite automata from any state delimited by a provided so-called context. Those cut sets are sets of automata local states such that disabling the activity of all local states of a cut set guarantees to prevent the reachability of the concerned local state. Automata networks are commonly used to represent the qualitative dynamics of interacting biological systems, such as signalling networks. The computation of cut sets can then lead to propose potential therapeutic targets that have been formally identified from the model for preventing the activation of a particular molecule.

The proposed algorithm works by propagating and combining the cut sets of local states along a *Graph of Local Causality* (GLC), that we introduce here upon automata networks. A GLC relates the local states that are necessary to occur prior to the reachability of the concerned local state. Several constructions of a GLC are generally possible and depend on the semantics of the model. We gave an example of such a construction for automata networks. That GLC has a size polynomial in the total number of local states in the network, and exponential in the number of local states within one automaton. Note that the core algorithm for computing the cut sets only requires as input a GLC satisfying a soundness property that can be easily extended to discrete systems that are more general than the automata networks considered here.

The computed cut sets form an under-approximation of the complete cut sets as the GLC abstracts several dynamical constraints from the underlying concrete model. Our algorithm does not rely on a costly enumeration of the potential sets of candidates, and thus aim at being tractable on very large networks.

A prototype implementation of our algorithm has been successfully applied to the extraction of cut sets from a Boolean model of a biological system involving more than 9000 interacting components. To our knowledge this is the first attempt of such a dynamical analysis for such large biological models. We note that most of the computation time is due to products between large sets of cut N-sets. To partially address this issue, we use of prefix trees to represent set of sets on which we have specialized operations to stick to sets of N-sets (Appendix A). There is still room for improvement as our prototype does not implement any caching or variable re-ordering.

The work presented in this paper can be extended in several ways, notably with a *posterior enlarging of the cut sets*. Because the algorithm computes the cut N-sets for each node in the GLC, it is possible to construct *a posteriori* cut sets with a greater cardinality by chaining them. For instance, let $kps \in \mathbb{V}(a_i)$ be a cut N-set for the reachability of a_i, for each $b_j \in kps$ and $kps' \in \mathbb{V}(b_j)$, $(kps \setminus \{b_j\}) \cup kps'$ is a cut set for a_i. In our biological case study, this method could be recursively applied until cut sets are composed of states of automata only acting for the environmental input.

With respect to the defined computation of cut N-sets, one could also derive *static reductions* of the GLC. Indeed, some particular nodes and arcs of the GLC can be removed without affecting the final valuation of nodes. A simple example are nodes representing objectives having no solution: such nodes can be safely removed as they bring no candidate N-sets for parents processes. These reductions conduct to both speed-up of the proposed algorithm but also to more compact representations for the reachability causality.

Finally, future work is considering the identification of necessary transitions (reactions), in addition to local states, that are necessary for a reachability to occur. The introduction of additional dynamical constraints in the GLC, such as conflicts or time scales, would also help to increase the number of cut sets identifiable by such abstract interpretation techniques.

Acknowledgements. LP and GA acknowledge the partial support of the French National Agency for Research (ANR-10-BLANC-0218 BioTempo project).

References

1. Bernardinello, L., De Cindio, F.: A survey of basic net models and modular net classes. In Rozenberg, G., ed.: Advances in Petri Nets 1992. Volume 609 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1992) 304–351
2. Kauffman, S.A.: Metabolic stability and epigenesis in randomly connected nets. Journal of Theoretical Biology **22** (1969) 437–467
3. Thomas, R.: Boolean formalization of genetic control circuits. Journal of Theoretical Biology **42**(3) (1973) 563 – 585
4. Richard, A.: Negative circuits and sustained oscillations in asynchronous automata networks. Advances in Applied Mathematics **44**(4) (2010) 378 – 392
5. Bernot, G., Cassez, F., Comet, J.P., Delaplace, F., Müller, C., Roux, O.: Semantics of biological regulatory networks. Electronic Notes in Theoretical Computer Science **180**(3) (2007) 3 – 14
6. Harel, D., Kupferman, O., Vardi, M.Y.: On the complexity of verifying concurrent transition systems. Information and Computation **173**(2) (2002) 143 – 161
7. Paulevé, L., Magnin, M., Roux, O.: Static analysis of biological regulatory networks dynamics using abstract interpretation. Mathematical Structures in Computer Science **22**(04) (2012) 651–685

8. Shier, D.R., Whited, D.E.: Iterative algorithms for generating minimal cutsets in directed graphs. Networks **16**(2) (1986) 133–147
9. Lee, W.S., Grosh, D.L., Tillman, F.A., Lie, C.H.: Fault tree analysis, methods, and applications - a review. IEEE Transactions on Reliability **R-34** (1985) 194–203
10. Tang, Z., Dugan, J.: Minimal cut set/sequence generation for dynamic fault trees. In: Reliability and Maintainability, 2004 Annual Symposium - RAMS. (2004)
11. Klamt, S., Gilles, E.D.: Minimal cut sets in biochemical reaction networks. Bioinformatics **20**(2) (2004) 226–234
12. Samaga, R., Von Kamp, A., Klamt, S.: Computing combinatorial intervention strategies and failure modes in signaling networks. Journal of computational biology : a journal of computational molecular cell biology **17**(1) (Jan 2010) 39–53
13. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1**(2) (1972) 146–160
14. de Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. Journal of Computational Biology **9** (2002) 67–103
15. Samaga, R., Saez-Rodriguez, J., Alexopoulos, L.G., Sorger, P.K., Klamt, S.: The logic of egfr/erbb signaling: Theoretical properties and analysis of high-throughput data. PLoS Comput Biol **5**(8) (08 2009) e1000438
16. Hinkelmann, F., Laubenbacher, R.: Boolean models of bistable biological systems. Discrete and Continuous Dynamical Systems - Series S **4**(6) (2011) 1443 – 1456
17. Klamt, S., Saez-Rodriguez, J., Gilles, E.: Structural and functional analysis of cellular networks with cellnetanalyzer. BMC Systems Biology **1**(1) (2007) 2
18. Paulevé, L., Magnin, M., Roux, O.: Refining dynamics of gene regulatory networks in a stochastic π-calculus framework. In: Transactions on Computational Systems Biology XIII. Volume 6575 of Lecture Notes in Comp Sci. Springer (2011) 171–191
19. Klamt, S., Saez-Rodriguez, J., Lindquist, J., Simeoni, L., Gilles, E.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics **7**(1) (2006) 56
20. Schaefer, C.F., Anthony, K., Krupa, S., Buchoff, J., Day, M., Hannay, T., Buetow, K.H.: PID: The Pathway Interaction Database. Nucleic Acids Res. **37** (2009) D674–9
21. Moustakas, A., Heldin, C.H.: Signaling networks guiding epithelial-mesenchymal transitions during embryogenesis and cancer progression. Cancer Sci. **98**(10) (Oct 2007) 1512–1520
22. Drabsch, Y., Ten Dijke, P.: TGF-β signalling and its role in cancer progression and metastasis. Cancer Metastasis Rev. **31**(3-4) (Dec 2012) 553–568

A Implementation of Sets of Minimal N-Sets

This appendix gives some details on the data structure we developed to efficiently manipulate sets of minimal N-sets, i.e., sets of N-sets containing no sursets. The data structure is similar to prefix trees, on which operations have been design to perform union, product and simplification (minimisation) of sets of N-sets. An OCaml[5] implementation of these routines is available at `http://code.google.com/p/pint/source/browse/pintlib/kSets.ml`.

Given a (possibly infinite) set of totally ordered elements, such as integers, the data structure is a forest where leafs are either $\perp$ or $\top$, and intermediate nodes are elements. Each path from any root to any leaf form a strictly increasing sequence of elements. The maximum height of the forest is $N+1$. Fig. 2 gives an example of such a structure.

$$\begin{tikzcd}
 & 1 \arrow[dl] \arrow[dr] & & 2 \arrow[d] \\
2 \arrow[d] & & 3 \arrow[d] & 5 \arrow[d] \\
4 \arrow[d] & & \top & \top \\
\top & & &
\end{tikzcd}$$

Fig. 2. Representation of set of sets $\{\{1,2,4\},\{1,3\},\{2,5\}\}$

Data structure An instance of the data structure is either $\top$, acting for the set containing the empty set ($\{\emptyset\}$), or $\perp$, acting for the empty set, or an (ordered) associative map from (prefix) elements to other instances of the data structure. This is summarised by the following definition:

$$\mathcal{D} ::= \top \mid \perp \mid \{i_1 \mapsto \mathcal{D}_1, \cdots, i_k \mapsto \mathcal{D}_k\} \tag{1}$$

with $k \geq 1$ and $i_1 < \cdots < i_k$. Given $\mathcal{D} = \{i_1 \mapsto \mathcal{D}_1, \ldots, i_k \mapsto \mathcal{D}_k\}$, $i_1, \ldots, i_k$ are the *prefixes*, and $\mathcal{D}_1, \ldots, \mathcal{D}_k$ their corresponding *suffixes*. We also note $\forall j \in [1;k], \mathcal{D}(i_j) \stackrel{\Delta}{=} \mathcal{D}_j$. As mentioned, $\perp$ acts as the empty set, so any prefix mapped to an empty set can be removed:

$$\{i_1 \mapsto \perp\} \equiv \perp$$
$$\{i_1 \mapsto \mathcal{D}_1, L, i_j \mapsto \perp, R\} \equiv \{i_1 \mapsto \mathcal{D}_1, L, R\}$$

Hereafter, we assume that this removing is done implicitly.

[5] `http://caml.inria.fr`

Example 3. The set of sets $\{\{1,2,4\},\{1,3\},\{2,5\}\}$ is encoded as

$$\{1 \mapsto \{2 \mapsto \{4 \mapsto \top\}, 3 \mapsto \top\}, 2 \mapsto \{5 \mapsto \top\}\}$$

which corresponds to the forest in Fig. 2.

We first describe two helper functions on top of $\mathcal{D}$ that will be used for the operations.

INDS*($\mathcal{D}$)* Given a data structure $\mathcal{D}$, the INDS function returns the sequence of prefix elements in the *reverse order*. If $\mathcal{D}$ is either $\top$ or $\bot$, the empty sequence is returned.

UP*($\mathcal{D}, h$)* Given a data structure $\mathcal{D}$ and a level h (initially 1), UP removes all the sets in $\mathcal{D}$ that can not be extended with one additional elements, i.e., all the sets with at least N elements.

```
function INDS($\mathcal{D}$)
    if $\mathcal{D} \in \{\top, \bot\}$ then return []
    else if $\mathcal{D} \equiv \{i_1 \mapsto \mathcal{D}_1, \cdots, i_k \mapsto \mathcal{D}_k\}$ then
        return $[i_k, \ldots, i_1]$
    end if
end function
```

```
function UP($\mathcal{D}, h$)
    if $\mathcal{D} \in \{\top, \bot\}$ then return $\mathcal{D}$
    else if $h \geq N$ then return $\bot$
    else
        for $i \leftarrow$ INDS($\mathcal{D}$) do
            $\mathcal{D}(i) \leftarrow$ UP($\mathcal{D}(i), h+1$)
        end for
        return $\mathcal{D}$
    end if
end function
```

Union and product The UNION and PRODUCT (realizing the $\tilde{\times}$ operator described in Sect. 1) operations guarantee the ordering between prefixes, and that no set has cardinality strictly greater than N.

UNION*($\mathcal{D}_a, \mathcal{D}_b$)* Given two set of sets, this function merges the prefixes of sets.

PRODUCT*($\mathcal{D}_a, \mathcal{D}_b, h$)* Given two set of sets at level h, the product is computed as follows: if two sets share the same prefix, the product results from the product of suffixes; if two sets have different prefixes, the one having the highest prefix is augmented by one level (if possible), and its product is computed with the suffix of the other. The result is the suffix of the lowest prefix.

```
**function** $\textsc{union}(\mathcal{D}_a, \mathcal{D}_b)$
    **if** $\mathcal{D}_a = \top$ **or** $\mathcal{D}_b = \top$ **then return** $\top$
    **else if** $\mathcal{D}_a = \bot$ **then return** $\mathcal{D}_b$
    **else if** $\mathcal{D}_b = \bot$ **then return** $\mathcal{D}_a$
    **else**
        $\mathcal{D} \leftarrow \mathcal{D}_b$
        **for** $i \leftarrow \textsc{inds}(\mathcal{D}_a)$ **do**
            **if** $i \in \textsc{inds}(\mathcal{D}_b)$ **then**
                $\mathcal{D}(i) \leftarrow \textsc{union}(\mathcal{D}_a(i), \mathcal{D}_b(i))$
            **else**
                $\mathcal{D}(i) \leftarrow \mathcal{D}_a(i)$
            **end if**
        **end for**
        **return** $\mathcal{D}$
    **end if**
**end function**
```

```
**function** $\textsc{product}(\mathcal{D}_a, \mathcal{D}_b, h)$
    **if** $\mathcal{D}_a = \top$ **then return** $\mathcal{D}_b$
    **else if** $\mathcal{D}_b = \top$ **then return** $\mathcal{D}_a$
    **else**
        $\mathcal{D} \leftarrow \bot$
        **for** $i \leftarrow \textsc{inds}(\mathcal{D}_a)$ **do**
            **for** $j \leftarrow \textsc{inds}(\mathcal{D}_b)$ **do**
                **if** $i = j$ **then**
                    $\mathcal{D}_{ij} \leftarrow \textsc{product}(\mathcal{D}_a(i), \mathcal{D}_b(j), h+1)$
                **else if** $i > j$ **then**
                    $\mathcal{D}_{a/i} \leftarrow \{i \mapsto \textsc{up}(\mathcal{D}_a(i), h+1)$
                    $\mathcal{D}_{ij} \leftarrow \textsc{product}(\mathcal{D}_{a/i}, \mathcal{D}_b(j), h+1)$
                **else if** $i < j$ **then**
                    $\mathcal{D}_{b/j} \leftarrow \{j \mapsto \textsc{up}(\mathcal{D}_b(j), h+1)$
                    $\mathcal{D}_{ij} \leftarrow \textsc{product}(\mathcal{D}_a(i), \mathcal{D}_{b/j}, h+1)$
                **end if**
                $\mathcal{D} \leftarrow \textsc{union}(\mathcal{D}, \mathcal{D}_{ij})$
            **end for**
        **end for**
        **return** $\mathcal{D}$
    **end if**
**end function**
```

Sursets simplification (minimisation) The above operations do not guarantee that the resulting set of N-sets is minimal, i.e., no sursets are present. Thanks to the ordering of elements in the forest, removing sursets of a given set can be done efficiently by only checking the sets having a lower prefix.

REMOVE*($\mathcal{D}_p, \mathcal{D}, h$)* Given a set of sets $\mathcal{D}_p$, this function removes all the sets in $\mathcal{D}$ that are sursets of sets in $\mathcal{D}_p$, starting at level h. If $\mathcal{D}_p$ is $\top$, $\mathcal{D}$ becomes the empty set; otherwise the process is repeated on all the suffixes having a prefix lower than the prefix of each set to remove.

SIMPLIFY*($\mathcal{D}, h$)* At level h (initially 1), ranging the prefixes from the higher to the lower, each prefixed set is recursively simplified, then any sursets of previously computed sets are removed from it. The outputted set of N-sets is hence minimal.

```
function REMOVE($\mathcal{D}_p, \mathcal{D}, h$)
    if $\mathcal{D}_p = \top$ then return $\perp$
    else if $\mathcal{D} \in \{\top, \perp\}$ then return $\mathcal{D}$
    else
        for $j \leftarrow$ INDS($\mathcal{D}_p$) do
            for $i \leftarrow$ INDS($\mathcal{D}$) do
                if $i = j$ then
                    $\mathcal{D}(i) \leftarrow$ REMOVE($\mathcal{D}_p(j), \mathcal{D}(i), h+1$)
                else if $i < j$ then
                    $\mathcal{D}_{p/j} \leftarrow \{j \mapsto$ UP($\mathcal{D}_p(j), h+1$)$\}$
                    $\mathcal{D}(i) \leftarrow$ REMOVE($\mathcal{D}_{p/j}, \mathcal{D}(i), h+1$)
                end if
            end for
        end for
        return $\mathcal{D}$
    end if
end function
```

```
function SIMPLIFY($\mathcal{D}, h$)
    if $\mathcal{D} \in \{\top, \perp\}$ then return $\mathcal{D}$
    else
        $\mathcal{D}' \leftarrow \perp$
        for $i \leftarrow$ INDS($\mathcal{D}$) do
            $\mathcal{D}_{/i} \leftarrow \{i \mapsto$ SIMPLIFY($\mathcal{D}(i), h+1$)$\}$
            $\mathcal{D}_{/i} \leftarrow$ REMOVE($\mathcal{D}', \mathcal{D}_{/i}, h$)
            $\mathcal{D}' \leftarrow$ UNION($\mathcal{D}', \mathcal{D}_{/i}$)
        end for
        return $\mathcal{D}'$
    end if
end function
```